―論文―
高岡短期大学紀要　第17巻　平成14年３月
Bull. Takaoka National College, Vol.17, March 2002

NANDゲートを用いた論理回路の教育用アプリケーションの開発

藤田徹也　常少英男＊　高松　衛＊　中嶋芳雄＊

要　旨

本稿では、NANDゲートのみの論理回路設計アプリケーションについて述べる。このアプリケーションは、ユーザインタフェースの強化によって、マウス操作のみで簡単に回路設計を行うことができる。

また、このアプリケーションではディレイを考慮に入れた出力のタイミング・シミュレーションによる検証が可能であり、より実際に近い形での論理回路設計が可能になる。

キーワード

論理回路設計、NANDゲート、タイミング・シミュレーション、ディレイ

1　はじめに

LSI設計は、古くからコンピューターの導入による設計自動化が推進された分野の一つである。LSIを製造する工程には、１ヵ月以上の長い期間と、それに伴う高いコストが必要である。つまり、製造後にLSIをテストして動作不良を見つけても、LSIはパッケージングされているために、実際の内部ゲート回路の動作を追跡してデバッグすることは難しい。さらには、設計ミスを見つけても、回路を修正して正常なLSIを再度製造するまでには再び長い期間と高いコストが必要である。そのため、設計ミスを低減し、再設計に入ることを極力避ける必要がある。よって、実際の製造工程に入る前に、コンピューターを用いて回路機能を可能な限り検証し、製造後に発生する可能性のある設計ミスをできる限り削減しなければならない。それには、比較的単純な作業では人手の加わる部分を少なくし、コンピューターによる設計作業の置き換えによる設計自動化を進める必要がある。

ここ最近のパソコンは高速CPUの登場により、ますますワークステーションとの境界がなくなりつつある。それに伴って、従来はワークステーションでしか動かせなかった各種のエンジニアリング・ツールが、身近なパソコンで動かせるようになってきている。回路図エディタやプリント基板CAD、ディジタルやアナログのシミュレーターのようなCAE(Computer Aided Engineering)においてもかなり実用になる製品が開発されている。さらにFPGAやPLDなどもパソコンレベルでも不自由なく設計から検証まで出来るようになってきている。

われわれはコンピューターを用いた回路設計について研究しており、代表的なものに２値論理回路をNANDゲートのみで設計する手法、MA法による回路設計が挙げられる。しかし、MA法を用いての回路設計では、回路設計はできるものの、コンピューターとのユーザーインタフェース部分がほとんど皆無な状態であるため、回路を設計していくにも大変手間と時間がかかってしまうという問題点があった。そこで、ユーザーインタフェース部分を制作し、誰もが簡単にMA法を用いて回路設計を行えるようにしたものが今回の研究である。

地域ビジネス学科　＊富山大学工学部

2　教育用アプリケーションの概要

2.1　機能

本アプリケーションは、NANDゲートのみの設計となっている。つまり、NANDゲート回路は最終出力側のゲートからみて、各段ごとにNANDゲートをORゲート、ANDゲートに置き換えることができるため、回路設計においては極めて一般的なものとみなされているからである。また、本アプリケーションは組み合わせ論理回路程度なら、基本的な設計からシミュレーションまで十分に行えるようになっており、さらに機能を最小限に抑えてあるため大変扱いやすい仕様となっている。

このようなことから、本アプリケーションは論理設計教育という見地から考えた場合には、有効なものであると思われる。つまり、高機能なCADほど実際の開発現場というシチュエーションでは非常に重宝されるが、操作を覚えるのにも多くの知識や時間が必要となってくるため、教育現場での使用には不向きと考えられるからである。その点、本アプリケーションは学部学生を対象とし、操作なども比較的簡単にできるように配慮されており、さらに設計した回路の検証並びに、シミュレーションなども視覚的に分かりやすいように制作されているので、教育用として最適なものであるといえる。

本アプリケーションの機能は以下の通りである。

- 16変数までのMA法を用いた回路設計
- 実際に回路が正常に動いているかを視覚的に分りやすくした検証機能
- ディレイを考慮に入れたタイミング・シミュレーション
- フリップフロップなど基本素子の学習プログラム

2.2　実装

本アプリケーションの本体はUNIX上のC言語で作成されており、GUIとしてTcl/Tkを利用している。

プログラム内では、各NANDゲートに対するデータ構造として以下の構造体を用いる。

```
struct nandnet{
  unsigned short eda[100];
    /*入力線が繋がっているゲートの番号*/
  unsigned char hensuu[16];   /*変数の有無*/
  unsigned char g_edasuu;   /*ゲートの枝数*/
  unsigned char v_edasuu;   /*変数の枝数*/
  }nanda[4000];

unsigned char hen;                /*変数の数*/
```

ゲートの番号がnandnet構造体のnandaと言う名前の配列の要素番号になり、メンバhensuu配列に変数の有無をＸ１から順に０，１，０，・・・というように格納しておく。プログラムの暴走を防ぐために入力ゲート間接続線の数と変数入力線の数を数えて、それぞれメンバg_edasuuとv_edasuuに格納しておく。変数の数は、変数henの中に格納しておく。この構造体では、ゲート数4000、16変数、最大入力ゲート間接続線数100の回路までの記憶領域を確保している。

また、このデータ構造に基づいて、回路を操作する基本関数を作成することができる。

```
void kenshou（int gn）{
          /* gnはゲート番号とする   */
  int  i;
  /***A***/
  for（i=0;i<nanda[gn].g_edasuu;i++）{
    if（nanda[gn].eda[０］ ==０）{
    /* 後ろにこれ以上ゲートがなければ */
      break;
    }
    kenshou（nanda[gn].eda[i]）;
    /*再帰的に呼び出す*/
```

```
        /***B***/
    }
    /***C***/
}
```

この関数は、引数gnで渡される番号のゲートになり、for文の中で入力ゲート間接続線に繋がっているゲートを次々と呼び出していく。このＣ言語のプログラムリストは、データで与えられた木を一巡するプログラムである。gnに出力ゲートの番号を渡して呼び出せば、木のデータを参照しながら、データの木の構造をつくる。Ａの部分で処理をさせれば出力ゲート側から、Ｃの部分で処理をさせれば入力端ゲート側から処理をして、関数の引数と返り値によって枝に沿って処理結果を渡していくことが出来る。また、Ｂの部分で処理をさせれば兄弟のゲート間での処理を行なうことが出来る。よって、Ａの部分には入力変数のみで出力を計算する処理を行わせ、また、Ｂの部分には入力接続線のみでゲートの出力値を計算する処理を行わせ、さらにＣの部分では、ＡとＢによって処理された計算値を返す処理を行えば、ゲートの出力値を求めることが出来るプログラムになることがわかる。従って、後述する論理回路の検証や、タイミング・シミュレーションが可能となる。

一方、Tcl/Tkでは、X-Windowにおける標準的なGUI部品がウィジェット(Widget)として提供されており、柔軟なGUIの構築が可能である。本アプリケーションでは、ウィンドウ操作に関連するウィジェットを利用することによって、データファイルからの回路図作成・表示、およびスクロール・拡大縮小等の機能の付加を、APIベースで作成する場合に比べてきわめて少ない工数で行うことができた。また、初期メニュー(NANDゲート数および変数の決定)、回路図上のメニューおよび検証画面の変数表示部分にスクリプトを組み込み、マウス操作によるイベント駆動を実現している。

3　論理回路図描写のための処理

今回の論理回路図表示プログラムでは、表示される回路図をより見やすくするために次の処理を加えている。

(1) スクロール機能の付加
(2) 拡大、縮小機能の付加
(3) 配置の最適化処理

(1)スクロール機能

ディスプレー上のウィンドウに回路図を表示したとき、回路規模が小さな回路であればウィンドウの枠内に収まり不都合は生じないが、回路規模が大きくなった場合には、回路がウィンドウの枠内には収まりきらず、はみ出して見えなくなるという問題がある。

この問題を解決するためには、大きく２つの方法が考えられる。一つは、回路図そのものを小さくし、全体を決められたウィンドウ内に映し出すという方法と、もう一つは画面をスクロールさせることによって、はみ出していた部分を見ていく方法である。前者については次のセクションに説明をのせるので省略して、後者のスクロール機能を用いる方法について説明する。スクロールを行うには、生成される回路図を直接ウィンドウに表示させるのではなく、回路図を一度メモリ内に描いておき、そこから必要な部分をウィンドウのほうへコピーしていけばよい(図１参照)。表示プログラムではこのようにして、スクロールを可能にしている。

この機能により出来上がった回路図がウィンドウ内からはみ出していた場合でも、回路図自体をスクロールさせることにより、はみ出している部分を自由に表示させることが可能になった。このことは回路図全体を見るために、ウィンドウ自体の大きさを変更する手間が不必要となった。(図２参照)

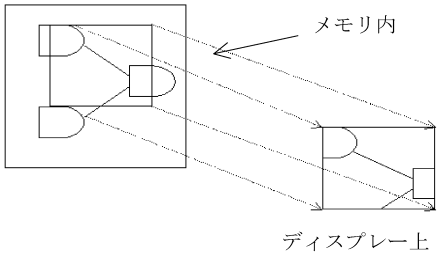
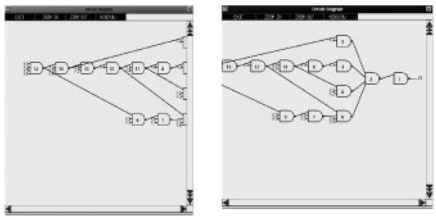


図１　コピー



（a）　　（b）

図２　スクロール

⑵拡大縮小機能

拡大縮小についてはゲートの大きさ、配置する場所などを共通のパラメーターを組み込んで制御することによって容易に行うことができる。

縮小機能によって、ウィンドウ内に収まりきらなかった大きな回路図の場合でも、回路図全体を縮小することで、ウィンドウ内に回路図を収めることができ、回路全体の様子を一目で把握できるようになった。また配線が多重に交差して見にくい場合や、シミュレーションを行う時にある一部分の動作をしっかり観察したい場合などには、この拡大機能が非常に役立つものとなる。（図３参照）

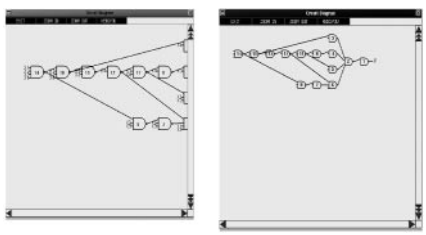


図３　拡大表示

⑶配置の最適化処理

ゲート間の配線の交差数がある規定数を越えた場合、ゲートの位置決めをする計算式を変更し、回路図を見やすくする。このために、プログラム内でゲートの位置決めが終わった後に、ゲート間の配線の交差数を求め、交差数がある規定の数を超えた場合には、ゲートのＹ座標を決める式を変更し配置決めをし直す。つまり、Ｙ座標は単純にrootからの枝の数で決められているので、その枝の数を場合によって大きく取ってやることで、ゲート間が離され見やすい回路図となる。この方法によりゲート間接続線が多重に交差しているときの回路図を図４に示す。

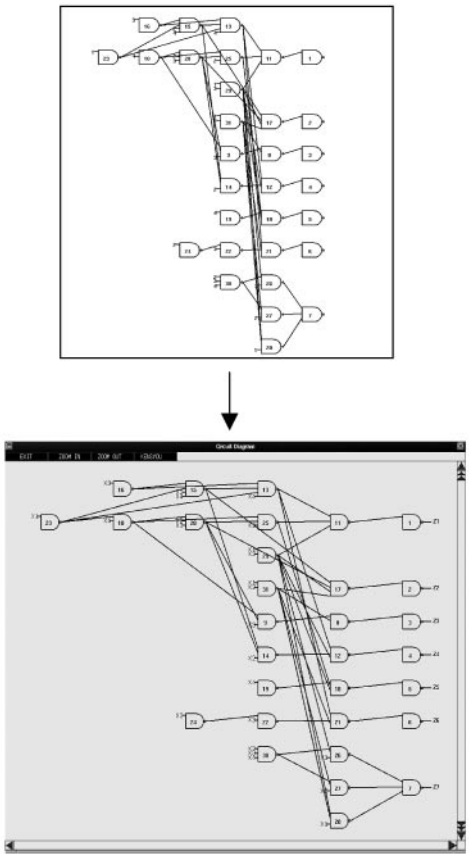


図４　配置の最適化処理

4　検証機能について

検証機能は出来上がった論理回路が真理値表どおりに動作しているかどうかを確認するためものである。また、そのときの各ゲートの出力も視覚的に分かるように、信号値が０ならば青色に、反対に信号値が１ならば赤色としてディスプレー上に表示するようになっている。(図５参照)

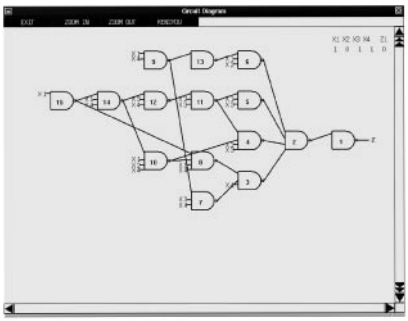


図５　検証画面

画面右上が現在の入力変数の値を示す。回路図で赤色の部分は電気信号が１(High)であるところを表し、青色の部分は０(Low)を表している。また、マウスを押していくごとに、入力変数が切り替わり、全ての入力変数の組み合わせについて検証していくようになっている。このように設計された回路の動作を視覚的に表現することで、回路の信号の様子をしっかりと認識することができ、論理回路を学ぶにも適している。

5　ディレイを考慮に入れた検証

４．の論理回路の検証プログラムは信号値である０と１が定常的に加わっているとしたときに、正しく動作しているかどうかを視覚的に確認するだけのものである。しかし、実際のデバイスでは時間の流れの中で動作しているので、それぞれのゲートごとに信号伝搬に伴うディレイ(遅延)やスレッショルドなどによる違いがみられ、そのため瞬間的に見てみると全く異なった回路になることがある。

そこでそのことを考慮に入れて、さらに応用をきかせたタイミング解析を行えるシミュレーションプログラムを制作した。このプログラムではタイミング・シミュレーションを行いながら、各ゲートの出力をタイミングチャート図に書き出す処理を行う。

5.1　ディレイについて

論理素子が演算動作をするには一定の時間を必要とする。これをゲートのディレイ(伝搬遅延時間)と呼び、入力端子から出力端子へ信号が到達するまでの時間をいう。したがって、種々の論理ゲート素子を結合した組み合わせ論理回路においては、論理をとるタイミングに注意しないと論理式上には現れない出力信号が発生し、回路が誤動作することがある。これをハザードという。このことを示すために以下の例(図６)で説明することにする。

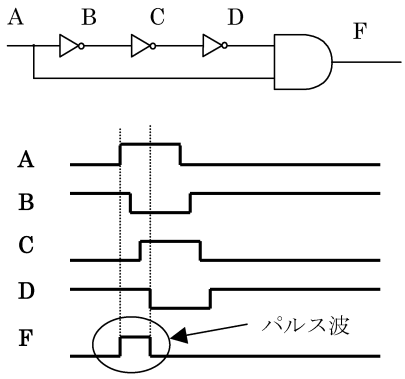


図６　ディレイによってできるハザード

図６の回路の出力は、論理的にはF=A・$\overline{A}$=０で意味のない回路である。しかし、実際にはNOTゲートの伝搬遅延時間により各段の信号は順次遅れて、出力側には図６に示すような細かいパルスが発生する。この回路は伝搬遅延時間を利用したワンショット回路として

しばしば用いられる実用的な回路である。ハザードを防止するためには、出力が安定してから出力信号を取り出すようにしなければならない。すなわち、回路の設計に際しては素子の伝搬遅延時間を考慮しておく必要がある。

5.2　タイミング・シミュレーションの方法

タイミングシミュレーションの方法について、以下の簡単な２変数の回路で説明する。(図７・図８参照)

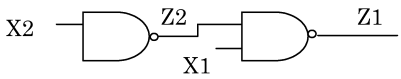


図７　簡単な２変数の回路

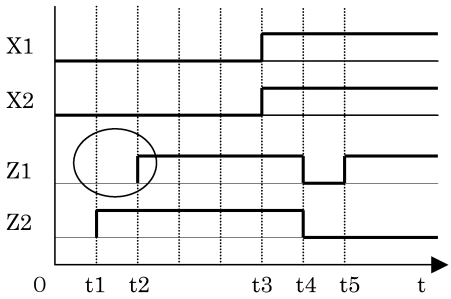


図８　タイミングチャート

⑴ｔ＝０

入力としてＸ１＝０、Ｘ２＝０を与える。

このときは、ゲートのディレイによりＺ１、Ｚ２はともに不明である。この不明という状態は、実際はほとんどが０(Low)になっていることが多いが、必ずしも０であるとは言い切れない。よって今回は初期状態を不明という形とする。

⑵ｔ＝ｔ１

Ｘ２＝０よりＺ２＝１となる。しかし、Ｚ１はt＝０のときのＸ１とＺ２の値で計算するのだが、このときのＺ２が不明の状態であったため、Ｚ１も不明である。

⑶ｔ＝ｔ２

このときはじめてt=t１のときの、Ｘ１とＺ２によって計算された値がディレイ分遅れてＺ１に現れ、最終的に安定した出力として求まる。

⑷ｔ＝ｔ３

⑶の状態からＸ１＝１、Ｘ２＝１と入力する。今度の場合は先ほどの値がｔ＝t３の段階でＺ１、Ｚ２ともに残っているため、Ｚ１＝１、Ｚ２＝１のままである。

⑸ｔ＝ｔ４

Ｘ２＝１よりＺ２＝０となるが、ディレイによって２番目のゲートではＺ２＝１のままなので、Ｚ１＝０となる。

⑹ｔ＝ｔ５

Ｚ２＝０、Ｚ１＝１となり、ここでようやく安定した状態となる。

以上がタイミングシミュレーションの流れである。図８にタイミングチャートを示す。

本アプリケーションでは、このタイミングシミュレーションの方法とは別に、不明な入力よりも０入力を優先する方法によってもシミュレーションが行えるようにした。なぜなら本アプリケーションはNANDゲートのみから構成されており、NANDゲートは一つでも０が入力されたなら必ず１を返すという性質を持つからである。

図７の同じ回路で説明すると、先の方法ではt＝０でＸ１＝０，Ｘ２＝０の入力信号を加えたとき、t=t１におけるＺ１の出力は不明の状態としていた。しかし、今回はより、ゲートの入力側に０が入った時点で他の入力が不明の状態でもＺ１を１と確定するようにした。(図９、10参照)つまり、この方法ではt＝t１のときにＺ１、Ｚ２とも１という結果となる。

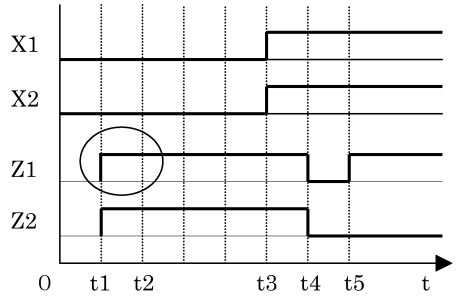


図９　０入力を優先した場合のタイミングチャート

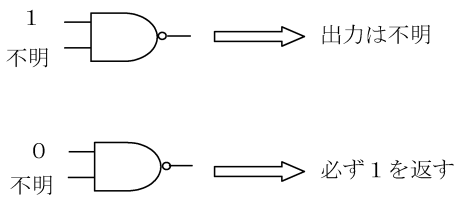


図10　ＮＡＮＤの計算

以上がディレイを考慮に入れたシミュレートの方法である。実際のシミュレーションでは、入力変数を入力したのち回路が安定したときには、それ以後の処理を打ち切るようにしている。そして、次の入力信号を得るために新しいウィンドウを開いて入力待ちの状態にする。この処理を行う理由とは、回路が安定した場合、その後の各ゲートの変化が無くなってしまうためであり、これ以後同じ処理を続けることは無駄な処理と判断したからである。

５.３　タイミングシミュレーションの流れ

図11にこのプログラムのフローチャートを図示する。

タイミングシュミレーションの流れ

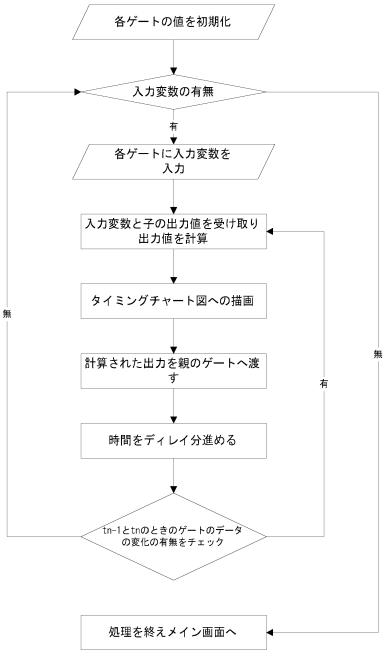


図11　プログラムのフローチャート

このプログラムでは、第４章の検証法と同じように、ゲートの信号の様子が視覚的に分かるようにしてある。つまり、入力変数を入力すると、その時点をｔ＝０として、タイミングチャート図に各ゲートの出力を描かせる。このとき同時に、回路図が表示されている画面にも、そのときのゲートの様子が分かるように、色分けされた回路図を写し出すことにしている。ここで、もし次の状態を知りたいならば、タイミングチャート図が表示されている画面上にマウスを移動させ、そこでクリックする。すると次の状態へと移り、その時点でのゲートの信号の様子とタイミングチャート図が画面に再び描かれるようになっている。こうしてマウスを押すたびに異なる時間ごとのゲートの信号の様子が段階をおって分かるようになっている。

さらに、このタイミングシミュレーションプログラムを利用した検証法も考えられる。

つまり、検証したい入力変数を入力し、その結果得られる出力を比べればよいのである。しかし、このプログラムを用いて検証を行っていくには、入力変数をマウスで地道に入力していかなければならず、これは大変面倒な作業となってくる。そこで今回新たに検証ボタンを押すだけで、全ての入力変数の組み合わせについて自動的に処理を行ってくれるプログラムを用意した。このプログラムにより瞬時に図12のような出力結果が得られ、簡単に検証が行えるようになっている。

図12に表示されている縦の波線部分は入力信号がちょうど変化する位置を表す。つまり、この線のちょうど左側のところが最終的な回路の出力となっている。よって回路を検証するには、この波線左側部分を参照していけばよいことになる。

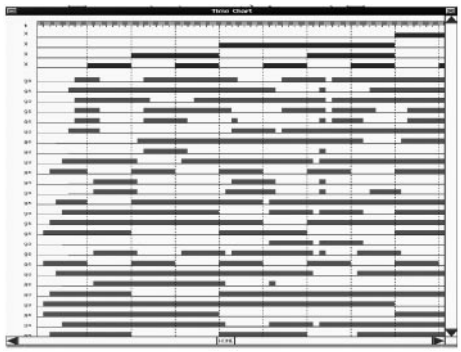


図12　タイミングチャート

6　7セグメントLEDのシミュレーション

教育用の設計の例として、７セグメントLED表示回路を構成するためのBCD７セグメントデコーダ回路をNANDゲートのみで設計する。ここで、MA法の初期回路生成プログラムは、１出力回路しか扱えないものを用いている。そこで多出力回路を設計するときは、一つの出力論理関数ごとに独立した初期回路を生成する。それを、一つの回路データファイルになるようにフォーマットを整えて整形することで多出力回路が生成される。

このような手順で表１のように表示するデコーダ回路を設計し、この回路で７セグメントLEDをシミュレーションしていく。図13が設計された回路と７セグメントの光る様子を表している。例えば、入力X１X２X３X４に００１０が入力されるとLEDに２と表示されるようになっている。

表１　７セグメントＬＥＤのための表

表示	入　力				出　　力						
	X1	X2	X3	X4	$\overline{Z_1}$	$\overline{Z_2}$	$\overline{Z_3}$	$\overline{Z_4}$	$\overline{Z_5}$	$\overline{Z_6}$	$\overline{Z_7}$
0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	1	1	0	0	1	1	1	1
2	0	0	1	0	0	0	1	0	0	1	0
3	0	0	1	1	0	0	0	0	1	1	0
4	0	1	0	0	1	0	0	1	1	0	0
5	0	1	0	1	0	1	0	0	1	0	0
6	0	1	1	0	0	1	0	0	0	0	0
7	0	1	1	1	0	0	0	1	1	1	1
8	1	0	0	0	0	0	0	0	0	0	0
9	1	0	0	1	0	0	0	1	1	0	0

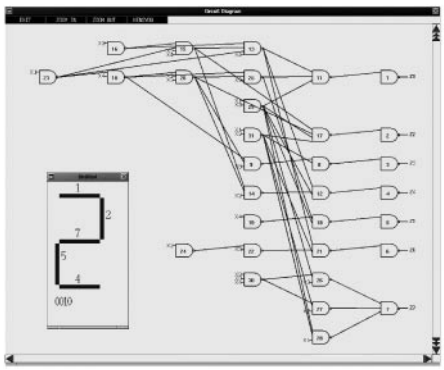


図13　７セグメントＬＥＤのシュミレーション画面

7　おわりに

本論文ではNANDゲートのみの論理回路設計アプリケーションについて述べてきた。制作されたアプリケーションは、以前のものよりユーザーインターフェース部分を強化してある。そのため、MA法での設計の知識が全

くない人でも、マウス操作のみで簡単に回路設計ができるようになった。さらに、このアプリケーションにはMA法による設計、検証のみでなく、基本的な論理素子の動作を学習できるプログラムや、７セグメントLEDの仕組みについても理解できるプログラムも用意してあるので、論理回路学習において大変役立てるものになっており、最初に述べている教育用ソフト不足の課題解消の一助となり得たものだと確信している。

参考文献

［1］羽根　孝泰：ドント・ケアを含む論理回路をNANDゲート回路で実現する一設計法、修士論文、TOYAMA-University（1997）

［2］秋山　敏行：論理回路の簡単化とグラフィックス表示プログラムの研究、修士論文、TOYAMA-University（1998）

［3］藤田　昌宏：トランスダクション法に基づく多段論理回路簡単化機能をもつ論理合成システムとその評価、情報処理学会論文誌、Vol. 30, No5, pp. 613-623（May1989）

［4］松田　秀雄、宮腰　隆、畠山　豊正：多段NANDゲート回路の一設計法、情報処理学会研究報告、設計自動化、DA-67

［5］秋山　敏行、松田　秀雄、山淵　龍夫、宮腰　隆、中嶋　芳雄：NANDゲート回路のグラフィックス表示、平成８年電気関係学会北陸支部連合大会

［6］常少　英男、松田　秀雄、中嶋　芳雄、山淵　龍夫、宮腰　隆：NANDゲート論理回路の教育用アプリケーション２、平成10年電気関係北陸支部連合大会

［7］常少　英男、松田　秀雄、中嶋　芳雄、山淵　龍夫、宮腰　隆：NANDゲートのみの回路の一設計法、平成９年電気関係北陸支部連合大会

Development of Educational Software for Designing Logic Circuit using NAND Gates

Tetsuya Fujita, Tsuneo Josho, Mamoru Takamatsu, Yosio Nakajima

We have developed the application software for designing logic circuit using only NAND gates. With reinforced user interface, it provides a simple way of designing a circuit by operating a pointing device.

Also, an output timing simulation for the purpose of verification, considering delay, enables a more practical logic circuit design.

Keywords

Logic circuit design, NAND gates, Timing simulation, Delay